Volume xx (200y), Number z, pp. 1–14

Geometry Images of Arbitrary Genus in the Spherical Domain

Mathieu Gauthier Pierre Poulin

LIGUM, DIRO, Université de Montréal

Abstract

While existing spherical parameterization algorithms are limited to genus-0 geometrical models, we believe a wide class of models of arbitrary genus can also benefit from the spherical domain. We present a complete and robust pipeline that can generate spherical geometry images from arbitrary genus surfaces where the holes are explicitly represented. The geometrical model, represented as a triangle mesh, is first made topologically equivalent to a sphere by cutting each hole along its generators, thus performing genus reduction. The resulting genus-0 model is then parameterized on the sphere, where it is resampled in a way to preserve connectivity between holes and to reduce the visual impact of seams due to these holes. Knowing the location of each pair of boundary components in parametric space, our novel sampling scheme can automatically choose to scale down or completely eliminate the associated hole, depending on geometry image resolution, thus lowering the genus of the reconstructed model. We found our method to scale better than other geometry image algorithms for higher genus models. We illustrate our approach on remeshing, level-of-detail rendering, normal mapping and topology editing.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Boundary Representations

1. Introduction

Geometry images [GGH02] offer an alternative representation for triangle meshes, that unifies the geometry and connectivity of an object in an image-like structure. In this representation each pixel (RGB) encodes the position (XYZ) of a single vertex and each group of four adjacent pixels forms a quadrilateral. To obtain the completely regular remeshing required to fit an object into an image, a series of cuts are first performed on the surface in order to open it in a disk-topology. The boundary vertices are then distributed on the boundary of a unit square and the interior is parameterized using an algorithm that minimizes geometric-stretch energy. Finally, the surface is resampled using a regular grid and each sampled position is stored in an image. This simple data structure implicitly encodes the connectivity, alleviates multiple random lookups required by the parsing of traditional mesh data structures, and is well suited to modern graphic hardware.

One of the difficulties with geometry images comes from the finite sampling resolution used to create the images. Traditional single-patch planar geometry images must ensure that each segment of the cut path, which opens the model into a disk-topology, is properly aligned with the sampling grid in parametric space. For higher genus surfaces, the complexity of the cut path can make it impossible to sample the geometry image below a given resolution. Moreover, these associated boundary constraints severely limit the efficiency of the parameterization for the interior vertices.

The same problem is observed in multi-patch systems, when patches of widely different sizes are generated. It can be impossible to properly resample the surface below a certain resolution without damaging the topology of the reconstructed surface, resulting in non-manifold surfaces.

One solution to this problem is to explicitly represent the holes in parametric space and to choose to sample them or not, based on the relevance of the holes in the input model and based on the desired image resolution. By doing so, it becomes possible to obtain a manifold reconstruction, but of a lower genus than the original surface.

Another problem with planar geometry images comes from the stretching and compression that occur when im-

posing artificial boundary constraints required to avoid visible seams. Cuts in the continuous surface or even divisions of the surface into multiple patches help to reduce some of these deformations, but can introduce many discontinuities between the previously attached pieces of the surface. By *discontinuities*, we mean pixels where fetching the neighbor requires jumping to another location in the image.

A solution consists in using a parameterization already more appropriate for certain classes of surfaces. As noted by Praun and Hoppe [PH03] and Sheffer et al. [SPR06], spherical parameterization appears in fact more natural for genus-0 surfaces. We believe this parameterization could also be beneficial for higher genus surfaces, and this paper demonstrates how to extend spherical parameterization to higher genus surfaces. We also believe that using this parameterization as an intermediate representation can help generating geometry images of higher quality and having desirable properties, such as short cut paths and fewer discontinuities in image space.

Nevertheless, even if all our empirical results appear to favor spherical parameterization, nothing prevents our technique from being applied in a very similar fashion to planar parameterization.

We propose a complete and robust pipeline to transform arbitrary genus surfaces in spherical geometry images. The process, illustrated in Figure 1, follows three stages: (1) use genus reduction and special treatment of holes to convert an arbitrary genus surface to a genus-0 surface; (2) use the progressive mesh encoding of the resulting genus-0 surface to robustly optimize the projection of surface vertices in the spherical domain; and (3) efficiently sample and reconstruct the surface in parametric space to produce a well-behaved 3D mesh, even in presence of holes.

The paper is organized as follows. Section 2 first presents some related work in three domains from which we build up our solution. Then the three stages are detailed their respective sections (Sections 3, 4, and 5). Section 6 analyzes some of our results and shows applications to illustrate the robustness of our approach, while Section 7 discusses some of its limitations. Finally, Section 8 revisits our contributions, and presents some directions for improvements.

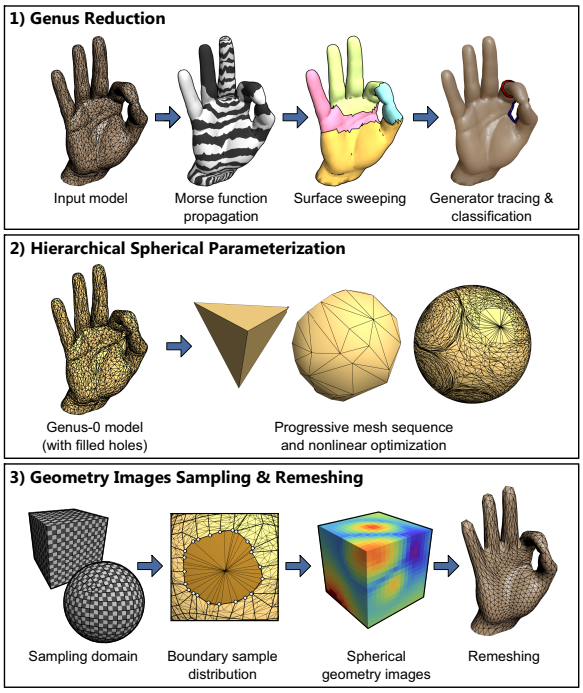


Figure 1: *Overview of the three stages of our method. An accompanying video [GP09a] illustrates each stage and shows various results.*

2. Previous Work

Since our work involves a complete pipeline, we will only and briefly survey the more closely related work in three main areas: spherical parameterization, geometry images, and computational topology. Their contributions will be presented in their respective subsections.

2.1. Spherical Surface Parameterization

Surface parameterization has received a lot of attention in the last decade. Its applications go far beyond texture mapping, for which it was originally developed. A wide variety of algorithms are now available, each trying to minimize angle or area distortion, or a combination of both. Although typically done in the planar domain, the spherical domain has received increased attention in the last years. Because the spherical domain is our primary parametric workspace, we will focus our attention on spherical surface parameterization. For sake of conciseness, we will avoid further discussion of a number of other related domains, such as quadrilateral meshing, globally smooth parameterization, periodic global parameterization, etc., and refer the interested reader to the thorough surveys of Floater and Hormann [FH05] and Sheffer et al. [SPR06], and the ALICE project of Lévy et al. [Lév07].

Although there is a good number of spherical parameterization algorithms available, they are not as mature as their planar counterparts. The problem originates from the inherent nonlinearity of the spherical domain and the lack of boundary conditions. In contrast to planar methods where the boundary is well defined and sometimes fixed, vertices on the sphere are free to move, thus increasing the number of possible solutions. Even worse, this lack of anchor points allows the parameterization to degenerate to the trivial solution where all vertices converge to the same point [FSD05].

To avoid such a degenerate configuration, Alexa [Ale99] projects the model on the sphere and maintains several

vertices fixed. Haker et al. [HAT*00] and Isenburg et al. [IGG01] first solve the problem in the planar domain and project it on the sphere. Gotsman et al. [GGS03] generalize the method of barycentric coordinates for planar parameterization to the sphere, but their method is limited to smaller meshes. Praun and Hoppe [PH03] use a progressive mesh sequence starting with a tetrahedron. Each new vertex is positioned at the barycenter of its neighbors and optimized to minimize the spherical geometric-stretch energy.

More recently, Friedel et al. [FSD05] study the divergence problems observed during spherical parameterization and propose a correction factor that allows the use of classical planar energies [Tut63, Flo97, SGSH02] to the spherical domain. This correction factor penalizes large triangles, making their energy grow to infinity as their surface moves closer to the center of the sphere.

2.2. Geometry Images

In their original paper introducing geometry images, Gu et al. [GGH02] demonstrate how it is possible to render a geometry image as several levels-of-detail by treating it as a mipmap hierarchy, and illustrate the power of its representation with the application of standard image processing algorithms such as image compression.

In the few following years, many extensions were presented, with applications for remeshing, ray tracing, compression, collision detection, shadow generation, etc. For conciseness sake, we will restrict ourselves to the different methods to generate geometry images rather than their applications.

Sander et al. [SWG*03] propose to reduce the distortion induced by the flattening on a disk-topology by building a texture atlas of geometry images, which they call *multi-chart geometry image*. Their algorithm grows a series of patches or charts on the surface of a geometric model and parameterize them individually. The charts are then sampled in a way that inter-patch connectivity is ensured and assembled in a single texture so that the amount of wasted pixels is kept to a minimum. This method effectively decreases the parameterization distortion at a higher complexity while introducing multiple discontinuities in the resulting geometry images.

Praun and Hoppe [PH03] propose an algorithm that parameterizes a genus-0 mesh on the sphere using a modified version of their geometric-stretch metric. They then resample the sphere using a sampling domain that can be unfolded into rectangular images. They experiment with a wide variety of domains such as Cube map, octahedron, and tetrahedron, and show better PSNR than traditional geometry images for certain classes of models.

When an object does not have a sphere topology, one can use multiple primitives to better cover the entire surface. The *PolyCube map* [THCM04] projects the object on a series of cubes approximating the shape of the object. They demonstrate how to project the surface on such an arrangement and explain how to sample the many different configurations that can arise between each cube. The external faces of each cube can then be used as geometry images with no wasted space. Unfortunately, in order to capture the shape of high genus surfaces, the number of cubes may become high.

Losasso et al. [LHSW03] propose to store the position of control points of a bicubic patch in the images. After parameterizing the object on the sphere and sampling it using an octahedral domain, they fit a bicubic patch on the surface while enforcing constraints so that boundaries close without creating visible seams. They then use modern graphic hardware to subdivide and render the surface.

2.3. Computational Topology

Morse theory [Mil63] has proven a valuable tool to describe the topology of a surface. The behavior of a differentiable function, called a Morse function, on a manifold can reveal the presence of critical points located where the gradient of the function vanishes. These points describe the structure of a surface. Originally developed for smooth surfaces, Morse theory extends naturally to piecewise linear functions on triangle meshes.

A good Morse function must be smooth and avoid creating degenerate, simultaneous, or artificial critical points, which would increase the complexity of the analysis. Hilaga et al. [HSKK01] compare the similarity between objects by using the geodesic distance as a Morse function to extract a hierarchical topological graph. This Morse function is invariant to rotation, but tends to create artificial maximum points on each protuberance of the surfaces.

Ni et al. [NGH04] present a *fair* Morse function that evolves between values assigned on constrained vertices set by the user. By solving a relaxed form of Laplace's equation, they force the values of all other vertices to be the average value of their neighbors.

Although Morse theory allows to identify the location of critical points on the surface, it fails to give a good global picture of its structure. The relationship between the critical points can be encoded in a topological graph called a *Reeb graph*. So far, a few algorithms to construct Reeb graphs of polygonal meshes have been introduced.

Zhang et al. [ZMT05] construct the Reeb graph of an object by sweeping the surface with a series of contours in the order specified by the Morse function. Each contour is split whenever it intersects itself, and two contours are merged when they collide. This method creates a graph easily embedded on the surface itself. More recently Pascucci et al. [PSBM07] build the graph from a stream of vertices and faces in arbitrary order, and keep the graph structure updated to reflect the new topology.

Finally, Wood et al. [WHDS04] repair topological errors in scanned models by analyzing the topology of their surfaces. After building the Reeb graph using an axis-aligned sweep, tiny handles (therein called holes) are found and removed using a volumetric approach. They show good result of surface simplification for commonly used models, and experiment with geometry images as well.

3. Genus Reduction

3.1. Overview

Our input geometrical model is an orientable 2-manifold, potentially with holes, defined by a mesh of triangles. The goal of our approach is to parameterize this triangle mesh in the spherical domain.

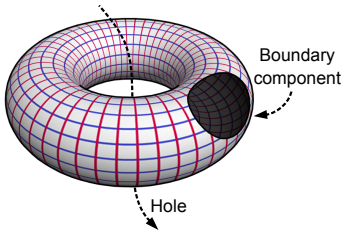


Figure 2: *Illustration of most of the topological terms used in this paper. The blue lines form the parallels, the red lines the meridians.*

In order to do so, the mesh must first become homeomorph to a sphere. Each existing boundary component (BC) is closed (or filled) by adding a new vertex at its barycenter and linking all of its edges to this barycenter, thus forming a fan (or cap) of new triangles. This fan-like configuration is star-shaped in the spherical parameterization that we will later create. This operation may generate triangles that intersect the mesh, or self-intersect, but this is not critical at this stage.

If the resulting mesh is genus-0, as computed with Euler's characteristic, we pass it directly for spherical parameterization (Section 4). Otherwise, we must convert it to a genus-0 mesh, which is the purpose of this section.

Genus reduction requires to identify a pair of generators for each hole. Since any genus-n manifold surface may be seen as a sphere with n tori connected to it, each hole can be opened by cutting the mesh along one of its two generators, then capped by a surface on each side of the cut, therefore each time reducing the genus by one. This is achieved by propagating a Morse function over the mesh, tracing and classifying the generators, as will be detailed in the following subsections.

3.2. Morse Function Propagation

To analyze the topology of our triangle mesh surface, we need a function to coherently traverse its surface. We adopted the fair Morse function from Ni et al. [NGH04], because it provides a smooth progression along the surface. However instead of propagating this Morse function over the mesh vertices, we propagate it over each triangle face. This results in a more coherent traversal of the surface during the sweep, treating one face at a time.

We use the linear formulation of the problem, and solve Laplace's equation using the *BiCGStab* solver [vdV92, BMM*07] to assign a value to each face. The result of this operation is a function that smoothly progresses between a minimum (-1) and maximum ($+1$) value face, and where the value of all other faces is computed as the average of the values from its three adjacent faces.

The surface is traversed by tracing contours on it. The contours along the edges of the mesh are traversed in order of the Morse function value, by adding the lowest-value unvisited triangle adjacent to the current contour. If a contour passes twice by one vertex, it is split into two contours. If two contours share one edge, it is merged into a single contour. Splitting and merging contours actually define a Reeb graph, even though we never create the actual graph. Our sweeping of the surface is similar to the one of Hilaga et al. [HSKK01].

3.3. Generator Tracing

The Reeb graph describes the complete topology of the mesh, but does not indicate precisely the location and shape of the generators for each hole. A merge event between two contours indicates the presence of a hole in the surface and the vertex where it happens is selected as the junction between the pair of generators. A generator is a circular list of connected edges that does not self-intersect. A hole is defined by one pair of generators that intersect at only one vertex.

We proceed with an approach similar to [SF03], but added two stages to increase robustness. We process each merge event between contours as they occur, and trace a pair of generators G_1 and G_2 defining the hole. We select the merge vertex v_m as the common vertex between the two. We use the shortest of the two merging contours as G_1 and trace G_2 through the visited values using an A* algorithm. If no paths can be found for G_2, we relax the condition and try again in all the faces while forcing the shortest path to leave from one side of G_1 and return from the other. In regions of high genus and low tessellation, it is possible that no paths can be found because of previously traced generators blocking the way. To avoid this situation, during the shortest path search, we keep a list of the blocking generators in the order that they were encountered. For each blocking generator, we perform a barycentric subdivision on the adjacent faces and try again to find a path. Barycentric subdivision can be seen as the combination of a 1-to-3 split on the face and a 2-to-4 split on the edges (Figure 3 (left)); it creates new ways through

which the path can pass (Figure 3 (center and right)). In rare situations, it is possible that a previously traced generator is passing by v_m. When this occurs, we retrace it and force it to avoid this vertex.

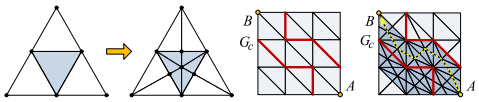


Figure 3: *(left) Barycentric subdivision of a triangle and its neighbors. (center) No possible path can traverse between points A and B without touching a vertex assigned to a previously traced generator (in red). (right) Subdividing all triangles adjacent to G_c allows a path to pass through.*

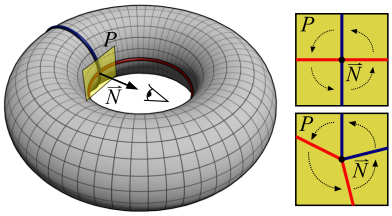


Figure 5: *To ensure that the normal at the intersection of two generators is valid, we look at the order of the projected generator edges on the tangent plane P. Projected view: (top right) valid configuration, (bottom right) invalid configuration.*

3.4. Generator Classification

Cutting a torus along one of its two generators opens it into a cylinder, and thus reduces the genus of a surface by one. Although choosing an arbitrary generator may work, it does not lead to the same parameterization quality. We propose to explicitly determine the types of generators and cutting along all generators of the same type. Each pair of generators is composed of one *meridian* and one *parallel*; we distinguish them using the knot-theory-based method of Steiner and Fischer [SF03].

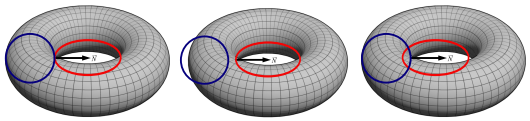


Figure 4: *Generator classification of Steiner and Fischer [SF03]. (left) A torus with its (red) parallel and (blue) meridian generator, and the normal at the joining vertex. Moving one of the generators in the reverse normal direction reveals its type. (center) If the loops become detached, the displaced generator is a meridian; (right) if they are nested, the displaced generator is a parallel.*

For each pair of generators, we translate one of them along the reverse surface normal direction, and compute the linking number on a valid projection plane. If this translation generates an *Hopf* link, the translated generator is a parallel, otherwise it is a meridian. The process is illustrated in Figure 4. One of the assumptions of this method is that the normal is a valid displacement direction. To determine if it is the case, we project the four edges of the two generators that are connected at v_m on the plane P perpendicular to the normal (Figure 5). We traverse the edges in clockwise order and make sure that no two edges of the same generator are next to each other. When this is not the case, we perform a series of angular perturbations to the normal in a limited range of angles to correct it.

Unfortunately, in extremely poorly tessellated areas, it is possible that no displacement normals can be found. In this case, we resort to an alternate strategy, and test if the barycenter of each generator is inside or outside the model. If one is inside and one is outside, we mark the first one as being a meridian and the second one as a parallel. If this last strategy fails as well, we assign an arbitrary type to one generator, and the other type to the other generator.

To show the robustness of our classification, for the genus-104 original Stanford Buddha model, the translation procedure was left with 42 unclassified generators, the perturbation of normal reduced this number to 8, and the interior test to 1. Therefore only one hole of the 104 original holes, residing on an almost flat region of the model, was assigned an arbitrary generator type. For the genus-55 original Stanford Dragon model, the corresponding numbers are respectively 18, 3, and 0, i.e. no generators were arbitrarily assigned.

3.5. Optimizations

Tracing the generators with the algorithm described earlier can produce overly complex generators. Since we will cut along only one of the generators of a given type, and discard the other, we can further optimize the generators of the type we are interested in. Our strategy is based on the observation that when all generators are traced, we can untrace some of them and retrace them in a different order, yielding another valid set of generators.

We process all the generators of the chosen type in decreasing order of their length. We retrace the generator using a shortest path algorithm and gather the list of generators of the other type that have blocked the path during the shortest path algorithm. We temporarily untrace each of them, one at a time, and retry to trace them to find a shortest path; we stop as soon as an improvement is obtained. We then retrace the untraced generators. After a few iterations of this procedure, we will have shortened the generators of the chosen type.

As a final step, for each generator of the chosen type, say

G_1, we take the associated generator of the other type, say G_2. For each vertex of G_2, we retrace a generator for G_1 with a shortest path algorithm, respecting all above constraints. We keep the overall shortest path for G_1.

All these operations are based on the intuition that smaller area holes should lead to less distortion in parametric space. While reducing the length of cut paths keeps the area of holes small for most convex holes, it is certainly not the case for concave holes. Therefore, for reasons similar to those which explain why it is very difficult to find the optimal cut path in planar geometry images or the optimal patch configuration in multi-patch solutions, finding the optimal configuration of generators leading to less parametric distortion is far from a trivial problem. Nonetheless, our shortest 3D cut heuristic in our implementation produced very good results so far.

3.6. Cutting and Capping

When all generators of the chosen type (parallel or meridian) have been traced and optimized, we cut the surface along the edges of each such generator. Each cut generates two boundary components (BCs), each of them capped by adding a new vertex at the barycenter of the boundary vertices and by connecting new triangles to this vertex. Each cut-and-cap process decreases by one the genus of the surface, to ultimately end up with a genus-0 surface that is suitable for spherical parameterization. Figure 6 shows the genus-8 Children model before and after genus reduction.

Figure 6: *The Children model before and after genus reduction. All parallel generators have been cut and capped with new faces.*

4. Spherical Parameterization

At this stage, our geometrical model with its closed (capped) holes is genus-0, and it is ready for spherical parameterization. We use a hierarchical version of the unconstrained spherical parameterization method presented of Friedel et al. [FSD05]. In their original paper, they simply project the model from its center of mass on the sphere, and let the solver find a bijective parameterization. This solution unfortunately diverged for models quite different than sphere-like, or at least, took a very long time to converge.

Instead, we adopted an hierarchical approach similar to the one of Praun and Hoppe [PH03]. We first build a sequence of progressive meshes of our model using the quadric error metric [GH97]. Since we have a genus-0 model, it reduces to a tetrahedron, which is easily distributed on the sphere. Then, at each iteration, we increase the number of vertices in the model by a certain factor (e.g. 2.0), place the added vertices at the barycenter of the kernel of their neighbors, and launch the unconstrained solver on this mesh. This approach proved quite efficient and stable, and generates no triangle flips during the entire optimization procedure.

Triangles used to close (cap) the holes are treated similarly to any regular triangle, except that we can multiply their area by a factor to increase or reduce their area for the solver. Figure 7 shows a few steps of this hierarchical solution for the triceratops model.

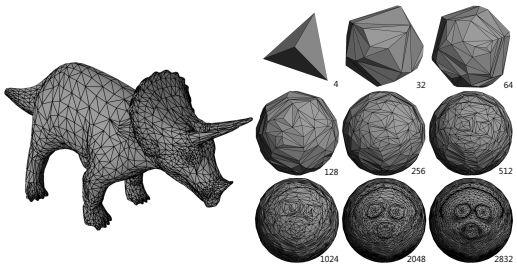


Figure 7: *Hierarchical spherical parameterization from a progressive mesh representation of the triangle mesh. The number of vertices is indicated next to its parameterization.*

This method has proven to be a stable, efficient, and intuitive way to parameterize a surface on the sphere. It worked very well on the more than 30 models we tested it on so far. Earlier in the project, we experimented with a few other spherical parameterization algorithms described in the literature, but failed to reach a sufficiently low energy minimum to guarantee a good sample distribution over the surface. Although our method is not guaranteed to reach a global minimum, the fact that it is essentially a gradient descent ensures that the energy will never increase and the use of first- and second-order derivatives helps to maintain a reasonable convergence speed. Just like any gradient descent, it requires a good initial guess in order to converge, and this is exactly what our hierarchical approach provides. From the base tetrahedron to the refined mesh, the algorithm always starts from a good initial solution that contains no triangle flips.

While spherical parameterization provides good results, as will be demonstrated in Section 6, it is worthwhile to mention that it is still possible before this stage to cut the sphere and map it into a planar parameterization. Our methods described in Sections 3 and 5 are not specific to spherical parameterization, although we use it to illustrate the more complex cases.

5. Sampling and Remeshing

5.1. Sampling Domain Creation

We call *sampling domain* any set of points that can be used to perform sampling of a parameterization. In the planar domain, sample points lie on a simple grid. In the spherical domain, sample points lie on the unit sphere. In order to allow the reconstruction of the surface, these points must contain an underlying notion of connectivity that allows the creation of triangles or quadrilaterals. Furthermore, if these samples are to be stored in geometry images, they must comply to some constraints, such as having exactly four neighbors, with the exception of boundaries where they may have two or three neighbors. Examples of sampling domains that fulfill these requirements are the Cube map, the Isocube, the HEALPix [GHB*05], and some unfoldable platonic solids such as the tetrahedron and the octahedron.

We explicitly instantiate the sampling domain by creating a quadrilateral mesh, where each vertex represents a sample. We associate to each sample one or more image positions where to copy the sample. While most samples will be copied in exactly one pixel of one image, samples on boundaries of multiple images will be copied in all of them. This redundancy allows for proper filtering and ensures completeness of each image in the sense that each individual geometry image contains all the information required to perform the remeshing.

When a sampling domain is not defined directly on the sphere, it is projected and optimized with the same parameterization algorithm described earlier. A more uniform distribution of samples is achieved when giving a higher weight to area distortion to the minimization process.

At this stage (Figure 8), the sampling domain can be superimposed on the spherical parameterization and sampling can be performed. However in our construction, the new triangles added to close holes during genus reduction should not be sampled. The problem is therefore to sample the spherical surface in presence of boundary components (BCs), while paying special attention to pairs of BCs generated from the same hole. To avoid creating open seams or T-vertices, we sample each side of a hole in a way that each side receives the same number of samples, and that each sample has the same position on both sides. To achieve this, we move the sample points to force them to lie on the BC, and only require moving domain vertices.

The algorithm proceeds in three steps:

1. Identify all samples located immediately around the BC, and discard all samples located inside the BC. (Section 5.2)
2. For the first BC of a pair, distribute the boundary samples on the BC perimeter. (Section 5.3)
3. For the associated BC of a pair, assign samples to the first BC to quilt the hole. (Section 5.4)

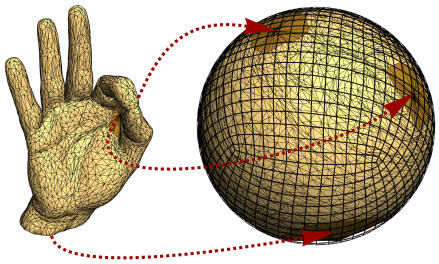


Figure 8: *(left) Geometrical model with three closed holes (BCs), one at the wrist and two between the joined thumb and index. (right) Triangle mesh of the model and of its closed holes transformed in the spherical parameterized space. The sampling domain is represented as a more regular grid.*

5.2. Identification of Boundary Samples

We process each hole by decreasing order of size. For each hole, we start by identifying domain samples located immediately around one of its two BCs. Similarly to a robot avoiding an obstacle by keeping it to its right, we walk around the BC in the sampling domain, testing intersection with the BC boundary. This is illustrated on the left side of Figure 9 and detailed below.

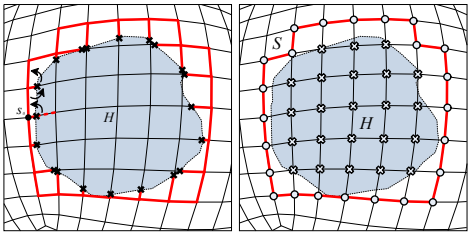


Figure 9: *(left) Traversal of the vertices of the sampling domain S outside the BC (H). The traversal starts at s_0 in the algorithm. (right) The vertices inside the BC are marked as null; the vertices on the immediate contour (in red) are marked as belonging to this BC.*

Let $H = \{h_0, h_1, ..., h_m\}$ be the set of edges around a BC in parameterization space. We want to identify the set of samples $S = \{s_0, s_1, ..., s_n\}$ located immediately around H. We start by locating one edge e_0 of the sampling domain that intersects one of the edges of H. Note that all intersection tests here are between arcs of great circles. We select the vertex of e_0 located *outside* of H as the starting sample s_0. We then iterate over the other neighbor samples connected to s_0 in counter-clockwise order starting from (but excluding) e_0, looking for an edge e_1 that does not intersect H. Its vertex, other than s_0, becomes vertex s_1. We then move to s_1, and look for edges in counter-clock order from edge e_1. We repeat this whole procedure until we close the loop and

reach again s_0. If the path reaches a dead end, i.e. where all unvisited edges intersect H, we back-track to a previous vertex s_{i-1} until the path can continue with a valid edge. In the case where it is impossible to circumvent H because of special cases of concavities of the BC, we try again with a new starting vertex for s_0.

The result of this procedure is a path on the sampling domain containing all samples of S immediately located around the BC. Samples located at the interior of this path are discarded by performing a flood-fill process. These samples are assigned null values in the associated geometry image pixels.

When walking around the BC in the sampling domain, we do not allow the path to move to a vertex immediately adjacent to a previously visited vertex, the only exception being the starting vertex s_0. This is illustrated in Figure 10. This simple rule is designed to leave a null sample inside very small holes or small protuberances. This will simplify the interpretation of the resulting geometry images and will be further discussed in Section 5.6. One of the consequences of this procedure is that the smallest configuration to represent a hole in the final image is a 3×3 with its central pixel being null.

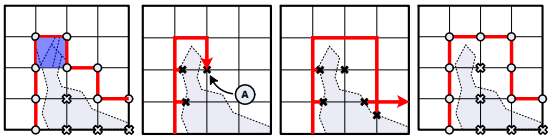


Figure 10: *Special cases for small holes or protuberances during the traversal of the sampling domain outside the BC. On the leftmost image, if we do not use the special constraint, the quadrilateral marked in blue would be wrongly generated, filling this part of the BC. On the three images on the right, our algorithm avoids this problem by identifying an adjacent vertex in A, and going around it.*

Once the samples S have been identified, they are marked as belonging to the BC and further BCs will not be able to cross or use theses samples. Moreover, during the walk, we do not allow the path to move to a vertex adjacent to any of the samples of the other BC of the pair, because it could potentially create degenerate quadrilaterals in the remeshing.

If the BC is paired with another BC, we process this other BC immediately afterward. The BC resulting in fewer domain samples is designated by H^- while the other is designated by H^+. Boundary samples of each BC will be referred to as S^- and S^+, respectively.

If there are not enough boundary samples available to circumscribe the BC, we try scaling the BC toward its center in parameterization space, using a predefined set of uniform scales (e.g. 0.5, 0.25, and 0.1). Ultimately if it still fails, we completely collapse the BC and its mate BC to a single point, effectively making both BCs invisible to any sampling. This feature, illustrated in Figure 11, is particularly useful when using low resolution sampling domains, for instance in a mipmap sequence. This is what gracefully and automatically reduces the genus of the reconstructed models as fewer samples are used.

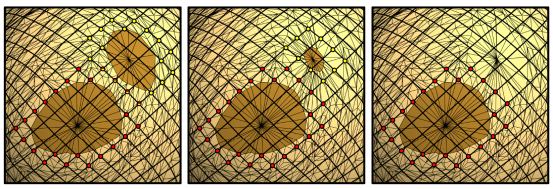


Figure 11: *(left) The larger hole being processed first, no valid set of samples can be found for the smaller hole. (center) Scaling the smaller hole allows the traversal algorithm to walk around it and find a valid set. (right) If for some reason no samples can be found, we collapse the hole (and its mate) into a single point. This automatically reduces the genus by one during the sampling process.*

5.3. Sample Distribution

Once the boundary samples have been identified for all the holes, we must align them with the silhouette of the BC. The goal is to distribute a limited number of samples to approximate the original shape of H. Many strategies can be used to perform this task, such as a uniform distribution around the perimeter, however since samples can sometimes be distributed unevenly around the BC, we believe it is better to consider the geometry in 3D during the distribution.

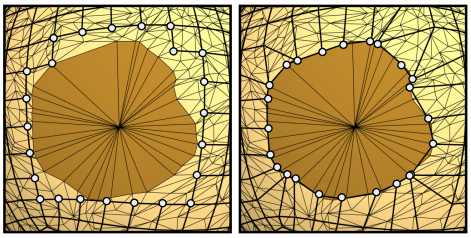


Figure 12: *The regular sampling grid and original triangles are displayed in the spherical domain. (left) The original BC with its boundary samples H on the grid are identified by white circles. (right) These samples are aligned and resorted around the edges of the BC.*

We locate for each sample s_i^- in 3D space the closest point p_i^- on the edges of H^-. We start by sampling the position and computing the 3D space position of each sample. This is done by ray casting and spherical barycentric interpolation in the sense of [Car07] (Section 3(c) of their

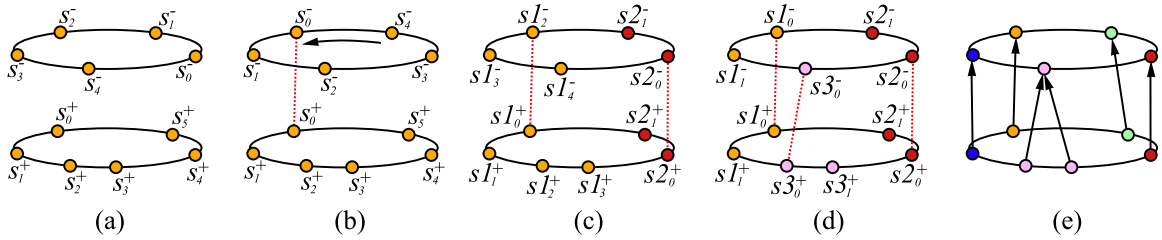


Figure 13: *(a) Two boundary components need to be quilted. (b) Aligning both sets so that s_0^+ is the nearest point to s_0^-. A dashed red line represents the pair of nearest points whenever it is possible, while respecting constraints. (c) Splitting S in two subsets because s_4^+ and s_3^- are the nearest pair of points (unless constraints applied). Recursion on $S2$ can stop immediately because both sets are of equal size. (d) Splitting $S1$ into two subsets. Recursion on $S1$ stops because both sets are of the same size. Recursion on $S3$ stops because there is only one element in $S3^-$. (e) Final sample assignment.*

paper). Because the resulting sequence of points P^- sometimes slightly differs from the order of the domain samples, we re-sort them and finally use these positions to compute the new parameterization-space positions of the samples S^-. The final result is illustrated in Figure 12.

5.4. Sample Assignment

The last stage, in this section, quilts the second side of the hole, and therefore only applies to BCs associated in a pair of BCs. We want to move each sample in S^+ in order to hermetically close the seam and avoid creating T-vertices. One way to accomplish this is to force each sample of S^+ to lie exactly on one of the samples of S^-, while respecting the order of the samples. Of course, there must be at least one s_i^+ associated to a given s_i^-. Since $|S^-| \leq |S^+|$, it is possible that multiple s_i^+ be associated to the same s_j^-, making it a surjective mapping. The quilting quality can be measured by summing the area of the quadrilaterals generated during the remeshing of the seam. Since there are $\binom{|S^+|-1}{|S^-|-1}$ different possible configurations, the optimal configuration computed by brute force is not practical as the difference between the number of samples in each set grows.

To obtain quality seam quilting without testing all possibilities, we adopt a divide-and-conquer strategy, illustrated in Figure 13. We start by rotating the order of the labels of the elements of S^+, so that s_0^+ is the closest point to s_0^- in 3D space. We then subdivide S^- in two half-subsets $S1^-$ and $S2^-$. We also subdivide S^+ in two half-subsets $S1^+$ and $S2^+$ so that $s2_0^+$ is the point closest to $s2_0^-$ in 3D space. While doing this, we impose the constraint that $|S1^-| \leq |S1^+|$ and $|S2^-| \leq |S2^+|$. We repeat this procedure recursively, and subdivide the resulting subsets in two, until $|Sn^-| = 1$ or $|Sn^-| = |Sn^+|$. In the first case, we assign to all elements of S^+ the positions of s_0^-, while in the second case, we assign $s_i^+ = s_i^-$. We repeat this procedure, starting from each s_i^- and choose the configuration generating the smallest remeshing area.

One must remember that when a hole is cut, the vertices of each created BC are in opposite orders in parameterization space. We must reverse one of the two sets before the procedure starts. Also when multiple samples s_i^+ are associated to a single s_j^-, a degenerated quadrilateral is created in the remeshing. A quadrilateral with two identical vertices is rendered as a triangle, and one with three identical vertices is safely discarded.

5.5. Spherical Sampling

Once the boundary samples have been processed, the sampling procedure is rather simple. A ray is shot from each domain vertex toward the origin, and the intersection point is computed. Any surface attribute, including position and normal, can be sampled and linearly interpolated using the spherical barycentric coordinates mentioned earlier. Each value is copied to the associated geometry image pixels.

5.6. Remeshing

The model is easily reconstructed from the geometry images. In the images, each group of four pixels $\{(x_i,y_j),(x_{i+1},y_j),(x_{i+1},y_{j+1}),(x_i,y_{j+1})\}$ represents a quadrilateral. If at least one of these pixels is null, the quadrilateral is discarded. It is for this reason that we intentionally leave a null pixel around a small hole or protuberance. Otherwise, the remeshing algorithm would require additional information to perform its task.

To generate a triangle mesh, we subdivide each quadrilateral into two triangles along the shortest diagonal. We also avoid generating triangles with an aspect ratio inferior or superior to limits specified by the user.

If other attributes, such as surface normals, are to be sampled at a higher resolution, we create a two-level hierarchical sampling domain and perform boundary sample identification and assignment using the lower resolution samples, and distribute the higher resolution ones in-between. For example, if we want to sample normals at a resolution four

times higher than positions, we create the sampling domain at the highest resolution and only consider every fourth sample as a *position* sample. The boundary sample distribution and assignment will be performed by only considering these samples and the higher resolution *normal* samples will be uniformly distributed in between.

6. Results

We implemented the complete algorithm in C++. We used an half-edge representation for all our mesh operations. The spherical parameterization algorithm was implemented in a similar fashion as Friedel et al. [FSD05]. The spherical energy functions were coded, differentiated, and converted into C++ using the automatic differentiation functions of *Maple* [Map]. We manually optimized and improved the robustness of our code to make significant gains. The various energy functions and their derivatives were then fed to one of the unconstrained solvers available in *TAO* [BMM*07]. We used the *NLS* solver, as it appeared to converge to faster results as the number of variables increases. As we perform splits in the progressive mesh sequence, almost null-area triangles may be created, so we set a minimum face area (10^{-8}) to keep the optimization process stable.

Our implementation is fully automatic and relies only on four mandatory user parameters : the domain type, the image resolution, and the angle/area weights for the spherical parameterization. In all our results, we selected the minimum and maximum face for the Morse function by using faces located at each end of one of the two longest bounding-box axes. We also chose to immediately discard all holes for which a BC did not overlap a domain edge. This way, we quickly eliminated all holes smaller (in parametric space) than the distance between two samples. We experimented with five sampling domains, namely, the Cube map, the Isocube, the HEALPix, the tetrahedron, and the octahedron.

Figures 14 and 15 illustrate the remeshing with two different sampling domains (HEALPix and Isocube) for a genus-1 model and a genus-8 model. The wireframe versions show the good distribution of vertices on the resampled surfaces.

Figure 16 illustrates the genus reduction of a model resampled at different resolutions, in a form of mipmap sequence. As the number of domain samples decreases, less and less holes can be represented in the spherical geometry images. This leads to similar results than the method of Guskov et al. [GKSS02], but with an automatic method.

As a comparison, we implemented the original geometry image algorithm from Gu et al. [GGH02]. We measured the PSNR [CRS98] of the surfaces reconstructed from the geometry images, with PSNR defined as $-20\log_{10}(h)$ where h is the Hausdorff distance with respect to the diagonal of the bounding box. Hence, the higher PSNR is, the better reconstructed the model is.

To parameterize the mesh in the plane, we experienced slow convergence with the geometric stretch parameterization algorithm [SSGH01]. A single pass of mesh refinement using local vertex neighborhood optimization converges very slowly, and is known to create artifacts such as *stretch marks* [YBS04], resulting in very elongated triangles in some portions of the parameterization. The fast geometric stretch algorithm of Yoshizawa et al. [YBS04] avoids these problems and is orders of magnitude faster, but it fails to reach a sufficiently low energy level to ensure a good distribution of samples on the surface, especially for large meshes.

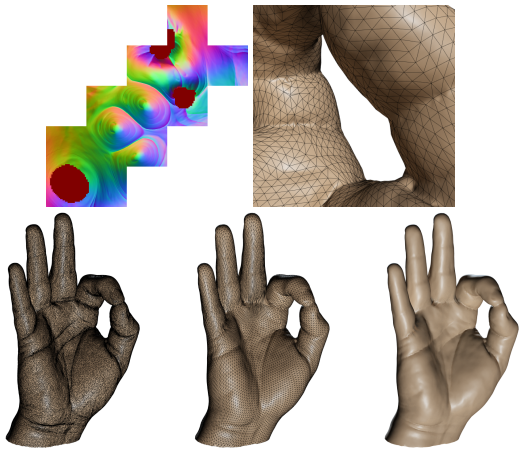


Figure 14: *From left to right. (top) HEALPix of normals and holes at a resolution of* $12 \times 129 \times 129$*; zoom in the region where the pair of BCs were cut. (bottom) Original model overlaid with its detailed mesh; spherical reparameterization of HEALPix domain at* $12 \times 33 \times 33$ *with overlaid wireframe; spherical reparameterization shaded (PSNR = 40.45).*

Figure 15: *From left to right. (top) Isocube map of* $6 \times 129 \times 129$*, and the original mesh with the generators. (bottom) Remeshing with the spherical parameterization, with wireframe overlaid and without (PSNR = 44.66).*

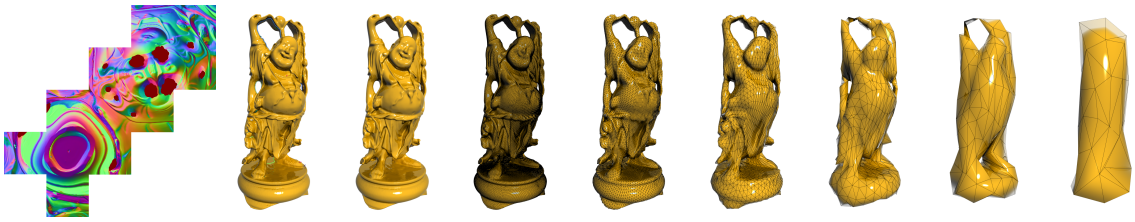


Figure 16: *From left to right: HEALPix of the normals of the reparameterized original model, original Stanford Buddha model, and resampling at seven different resolutions for each HEALPix image:* 129^2 *(PSNR = 42.31),* 65^2, 33^2, 17^2, 9^2, 5^2, *and* 3^2. *The resulting models have genus: 11, 9, 9, 7, 4, 2, and 0, respectively.*

To overcome these problems, we used a solution very similar to the unconstrained spherical parameterization algorithm used in our method.

Friedel et al. [FSD05] have shown that the geometric stretch energy is in fact a combination of angle and area distortions

$$E_{stretch}(f) = \begin{cases} \infty & \text{if } area(f_D) \leq 0 \\ \frac{area(f_M)^2}{area(f_D)^2} \cdot E_{Dirichlet} & \text{otherwise.} \end{cases} \quad (1)$$

where triangle f is represented as triangle f_M from the original mesh, or as triangle f_D in the parameterization domain. This type of formulation is not practical for use inside a black box nonlinear solver because it does not vary smoothly, as it reaches infinity whenever a triangle flips in the parameterization space. Instead, we use a 2D version of the combined energy function suggested by Friedel et al. [FSD05], which is a simple linear combination of the angle and area distortions

$$E_{combined} = a \cdot E_{Dirichlet} + b \cdot E_{area}. \quad (2)$$

Therefore we basically use the same overall parameterization technique in both planar and spherical pipelines, and also use similar area and angle weights. In all our results we used the same area and angle weights. We found the best configuration to be a high area preservation weight (e.g. $b = 1.0$) and a lower angle preservation, while never setting it completely to zero (e.g. $a = 0.01$).

We generated spherical geometry images of three common models using different sampling domains with approximatively the same number of pixels as the original method. For every model, we display the planar and spherical parameterizations, show the respective generated geometry images, and evaluate the encoding quality with the PSNR and display of a checkerboard texture pattern. We also give the lengths of the cuts, for the cut graphs and the generators, performed on the surface with each method.

To generate the planar geometry images for the Stanford Buddha and Cheese models, we used the surface growing algorithm described by Gu et al. [GGH02] to find the initial cut network. For the Children model, we manually selected the cut path, so that it would be near optimal.

In every case, the PSNR from our spherical geometry images was higher (remember that PSNR is a logarithmic scale) and the remeshings and parameterizations were visually smoother, even though a few hundred pixels are wasted in each image to represent holes. The created spherical parameterizations exhibit fewer regions of high stretch and compression than their planar equivalent. These results are summarized in Figure 17. The holes are drawn in uniform medium red (128,0,0) and the percentages of pixels they occupy are indicated between parentheses.

Because cut paths of single-patch planar geometry images must include all pairs of generators and all of them must be linked together, they do not scale very well on high genus surfaces, as they quickly become very long and put a real strain on the parameterization. Moreover, since all mate cut segments must be allocated the same length on the border of the image, this restricts the efficiency of the parameterization of the interior which may lead to poor sample distribution. For example, the generating a cut path of the Stanford Buddha using the algorithm of Gu et al. [GGH02] created 576 cut segments (1152 including mates) which means geometry images of resolution lower than 288×288 cannot be generated, in which case all segments will be allocated a length of 1 pixel regardless of their actual length. Our method can handle any genus surface. Lowering the resolution will automatically try to represent the larger holes and reduce the genus when it is impossible to do so. Also, our parameterization is completely independent of the sampling process. Resampling at a different image resolution does not require to reparameterize the mesh.

To illustrate that our spherical geometrical images can support all kinds of image processing algorithms, we implemented a simple Laplace filling algorithm to replace the hole between the thumb and the index of the Hand model.

In another example, we used a similar idea to create a hole in the palm. A user drew two pixel contours in parametric space, one located on each side of the palm. After discarding the pixels inside and around each contour, we moved each contour to the average position of all their vertices. Af-

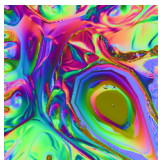
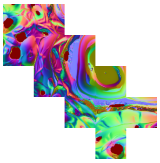
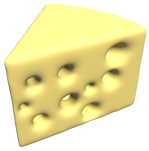
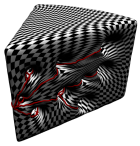
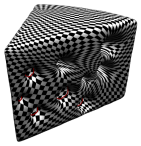
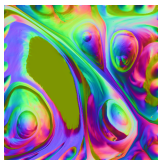
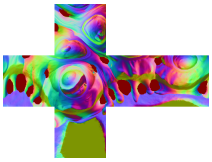
Original Model	Planar Param.	Spherical Param.	Planar Geom. Image	Spherical Geom. Image
Buddha 49990 vertices, genus-6	Cut path length : 4983.55	Generators length : 1324.79	 222 × 222 PSNR : 41.61	 65 × 65 × 12 HEALPix PSNR : 43.81 (4% null pixels)
 Cheese 56391 vertices, genus-9	 Cut path length : 105.53	 Generators length : 26.08	222 × 222 PSNR : 40.29	65 × 65 × 6 Cube map PSNR : 40.52 (3% null pixels)
Children 100000 vertices, genus-8	Cut path length : 114.40	Generators length : 77.00	 444 × 444 PSNR : 44.70	 129 × 129 × 6 Isocube PSNR : 46.08 (7% null pixels)

Figure 17: *Comparison between traditional geometry images and our arbitrary genus spherical geometry images.*

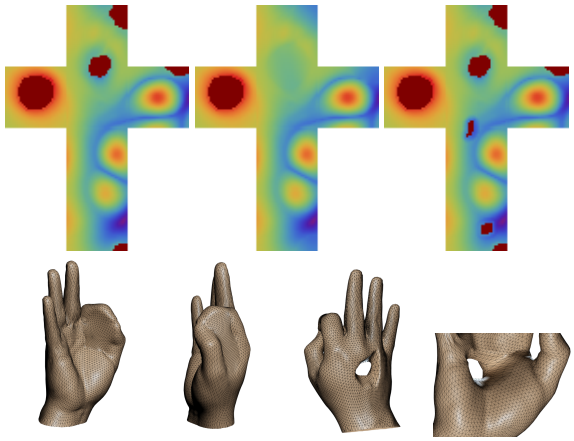


Figure 18: *From left to right. (top) Cube maps of the reparameterized original model, filling the hole between the thumb and index, and digging a hole in the palm drawn on the Cube map. (bottom) Two views of the filled hole, and two views of the dug hole.*

ter a few (3) relaxation steps, we snapped the vertices of each contour in a way to minimize the total vertex displacement. A contour relaxation step moves each vertex toward the average position of their immediate neighbors on the contour. Finally, Laplace filling was performed around each contour to join the two sides of the hole. Results for these two examples are shown in Figure 18.

7. Limitations

While we found the spherical domain to be appropriate for a wide variety of models we tested it on, it requires the input model to be *somewhat* spherical. Hence, even if our algorithm can easily process the Stanford Dragon, its elongated shape prevents it from fully benefitting from the spherical domain. The spherical parameterization of Figure 19 reveals that the middle section of the body of the dragon contains more elongated triangles than the head and the tail.

Another limitation comes from the fact that we not only represent where the surface is, but also where it is *not*. This means that holes must be relatively small compared to the rest of the surface in order to fully benefit from the method. Large holes will translate into more null pixels in the generated images. This problem can be alleviated by scaling down the triangles capping the holes in parameterization space, but this comes at the price of a more approximate reconstruction of the boundary of the hole. Some stretching and discontinuities can also occur due to the sampling and quilting between BC pairs with very different aspects after relaxation. While

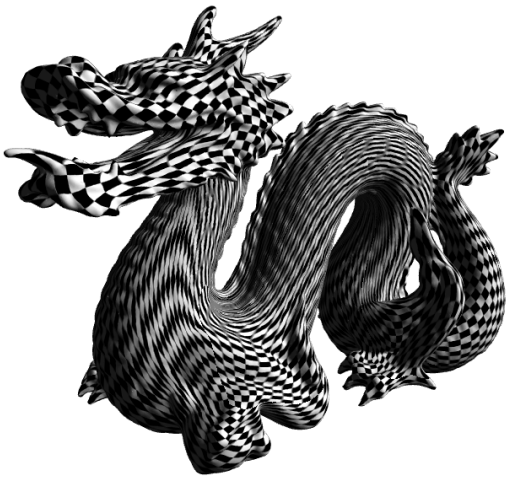


Figure 19: *Parameterizing the Stanford Dragon on the sphere creates regions of higher stretch in the middle section of the body.*

our results generally show good behavior in these regions, we cannot guarantee the absence of such discontinuities.

Finally, our method inherits many of the limitations of geometry images, one of them being the inability to preserve sharp features of the surface. Regular sampling in parametric space will create poor reconstructions when the domain samples are not precisely aligned with sharp features present on the surface. While realigning them to the grid can improve their reconstructions [GP09b], yet some limitations remain.

8. Conclusion and Future Work

We introduced a complete pipeline that generates geometry images for arbitrary genus surfaces at any given resolution. We use the spherical parameterization as an intermediate representation, where the holes in the surface are explicitly represented. We showed that unlike planar parameterization, spherical parameterization does not add artificial boundary constraints and is quite appropriate for generation of geometry images, even in the presence of holes. Our methodology has proven to be robust for all surfaces we tried it on.

We also improved on the efficiency of a number of critical stages in the code. We were able to reparameterize the Stanford Buddha and Dragon models from their original scanned representations, even though they contained topological anomalies. Our method compared favorably with our implementation of the traditional geometry images from Gu et al. [GGH02].

Among a number of our contributions, we improved the tracing and classification of generators, made hierarchical the unconstrained spherical parameterization method, and developed a complete remeshing algorithm for spherical parameterizations with BCs.

Although we showed that genus reduction works well for surfaces with holes ranging from tiny to large, we believe that the more natural match of genus-0 surfaces with spherical parameterizations should extend to genus-*n* surfaces with default (e.g. *n*-torus) parameterizations. More thorough experiments would be required to determine at which size a hole should be filled with genus reduction, or treated as a higher genus surface.

Cutting all parallel or all meridian generators seems to produce the best results. We need to develop a better insight into why that is, and if it is not always the case, to find out how to select the best generator for each hole configuration.

Finally, it would be interesting to compare the quality of our generators with the recent method of Dey et al. [DLSCS08], based on persistent homology, even though the latter requires the additional construction of the mesh interior.

9. Acknowledgements

The authors thank Neil F. Stewart and the anonymous reviewers for their detailed suggestions. This work was supported by grants from NSERC.

References

[Ale99] ALEXA M.: Merging polyhedral shapes with scattered features. In *SMI '99: Proc. Intl Conf. on Shape Modeling and Applications* (1999), pp. 202–210.

[BMM*07] BENSON S., MCINNES L. C., MORÉ J., MUNSON T., SARICH J.: *TAO User Manual (Revision 1.9)*. Tech. Rep. ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2007. http://www.mcs.anl.gov/tao.

[Car07] CARFORA M. F.: Interpolation on spherical geodesic grids: A comparative study. *J. Comput. Appl. Math. 210*, 1-2 (2007), 99–105.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (June 1998), 167–174.

[DLSCS08] DEY T. K., LI K., SUN J., COHEN-STEINER D.: Computing geometry-aware handle and tunnel loops in 3d models. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (2008), p. 45.

[FH05] FLOATER M. S., HORMANN K.: Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling* (2005), Springer, pp. 157–186.

[Flo97] FLOATER M. S.: Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design 14*, 4 (1997), 231–250.

[FSD05] FRIEDEL I., SCHRÖDER P., DESBRUN M.: Unconstrained spherical parameterization. In *ACM SIGGRAPH 2005 Sketches* (2005), p. 134.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. *ACM Trans. Graph. 21*, 3 (2002), 355–361.

[GGS03] GOTSMAN C., GU X., SHEFFER A.: Fundamentals of spherical parameterization for 3d meshes. In *SIGGRAPH '03* (2003), pp. 358–363.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97* (1997), pp. 209–216.

[GHB*05] GÓRSKI K. M., HIVON E., BANDAY A. J., WANDELT B. D., HANSEN F. K., REINECKE M., BARTELMANN M.: HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. *The Astrophysical Journal 622* (Apr. 2005), 759–771.

[GKSS02] GUSKOV I., KHODAKOVSKY A., SCHRÖDER P., SWELDENS W.: Hybrid meshes: multiresolution using regular and irregular refinement. In *Proc. Symposium on Computational Geometry* (2002), ACM, pp. 264–272.

[GP09a] GAUTHIER M., POULIN P.: http://www.iro.umontreal.ca/labs/infographie/papers/-Gauthier-2009-GIAGSD/, Jan. 2009.

[GP09b] GAUTHIER M., POULIN P.: Preserving sharp edges in geometry images. In *Proc. Graphics Interface '09* (May 2009).

[HAT*00] HAKER S., ANGENENT S., TANNENBAUM A., KIKINIS R., SAPIRO G., HALLE M.: Conformal surface parameterization for texture mapping. *IEEE Trans. Visualization and Computer Graphics 6*, 2 (2000), 181–189.

[HSKK01] HILAGA M., SHINAGAWA Y., KOHMURA T., KUNII T. L.: Topology matching for fully automatic similarity estimation of 3D shapes. In *SIGGRAPH '01* (2001), pp. 203–212.

[IGG01] ISENBURG M., GUMHOLD S., GOTSMAN C.: Connectivity shapes. In *VIS '01: Proc. Conf. on Visualization '01* (2001), pp. 135–142.

[Lév07] LÉVY B.: *Parameterization and deformation analysis on a manifold*. Tech. rep., http://alice.loria.com/, 2007.

[LHSW03] LOSASSO F., HOPPE H., SCHAEFER S., WARREN J.: Smooth geometry images. In *Proc. Eurographics/ACM SIGGRAPH symposium on Geometry Processing* (2003), pp. 138–145.

[Map] MAPLE: http://www.maplesoft.com/.

[Mil63] MILNOR J. W.: *Morse Theory*, vol. 51 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, USA, 1963.

[NGH04] NI X., GARLAND M., HART J.: Fair Morse functions for extracting the topological structure of a surface mesh. *ACM Trans. Graph. 23*, 3 (2004), 613–622.

[PH03] PRAUN E., HOPPE H.: Spherical parametrization and remeshing. *ACM Trans. Graph. 22*, 3 (2003), 340–349.

[PSBM07] PASCUCCI V., SCORZELLI G., BREMER P.-T., MASCARENHAS A.: Robust on-line computation of Reeb graphs: simplicity and speed. *ACM Trans. Graph. 26*, 3 (2007), 58.

[SF03] STEINER D., FISCHER A.: Explicit representation of object holes in topological and cut graphs. In *The 4th Israel-Korea Bi-National Conf.* (2003).

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Proc. Eurographics Workshop on Rendering* (2002), pp. 87–98.

[SPR06] SHEFFER A., PRAUN E., ROSE K.: Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (2006), 105–171.

[SSGH01] SANDER P. V., SNYDER J., GORTLER S. J., HOPPE H.: Texture mapping progressive meshes. In *SIGGRAPH '01* (2001), pp. 409–416.

[SWG*03] SANDER P. V., WOOD Z. J., GORTLER S. J., SNYDER J., HOPPE H.: Multi-chart geometry images. In *Proc. Eurographics/ACM SIGGRAPH symposium on Geometry Processing* (2003), pp. 146–155.

[THCM04] TARINI M., HORMANN K., CIGNONI P., MONTANI C.: Polycube-maps. *ACM Trans. Graph. 23*, 3 (2004), 853–860.

[Tut63] TUTTE W. T.: How to draw a graph. *Proc. London Math. Soc. 13* (1963), 743–767.

[vdV92] VAN DER VORST H. A.: BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput. 13*, 2 (1992), 631–644.

[WHDS04] WOOD Z., HOPPE H., DESBRUN M., SCHROEDER P.: Removing excess topology from isosurfaces. *ACM Trans. Graph. 23*, 2 (2004), 190–208.

[YBS04] YOSHIZAWA S., BELYAEV A., SEIDEL H.-P.: A fast and simple stretch-minimizing mesh parameterization. In *SMI '04: Proc. Shape Modeling International* (2004), pp. 200–208.

[ZMT05] ZHANG E., MISCHAIKOW K., TURK G.: Feature-based surface parameterization and texture mapping. *ACM Trans. Graph. 24*, 1 (2005), 1–27.